RIVET

Current status and future plans

Leif Lönnblad

Department of Theoretical Physics
Lund University

HERA/LHC workshop
DESY 07.03.14

Outline

Introduction

- Cedar
- RIVET
- RIVETGUN

The RIVET structure

- `RivetHandler`
- The `Analysis` base class
- Projections

Current Status

Cedar

RIVET is a part of the Cedar project containing several components.

- HEPDATA: a reimplementation of the Durham database of experimental results in HEP.
- RIVET: a C++ replacement of the HZTOOL library
- RIVETGUN: General interface to event generators to be used together with RIVET

- HEPFORGE: A repository for developing HEP code, including subversion and the Trac bug tracking system and Wiki.
- HEPML: an XML description of HEPDATA measurements and event generator steering.
- JETWEB: Web-based interface to RIVET, RIVETGUN and HEPDATA
- . . .

Rivet

- A C++-replacement for HZTool
- Includes analysis routines for comparison of event generators to measurements in HEPData
- Only for data corrected to particle level.
- Also includes utilities: jet finders, thrust calculations, . . .
- Should be completely generator-independent

RIVETGUN

- The functionality of HZSTEER
- Common interface to any Fortran or C++ event generator.
- Steered from HEPML setup files
- Produces HEPMC event objects which are sent to RIVET for analysis.

The RIVET structure

RIVET is a class library containing classes corresponding to a given experimental analysis.

Just as HZTOOL had one Fortran function per experimental paper, RIVET has one analysis class per paper.

The input to RIVET is HEPMC event objects, and the output is histograms (any AIDA-compliant histogram package can be used)

RivetHandler

```
// Create a RivetHandler given an analysis factory
RivetHandler RH(*MyHistoPackage::createFactory());

// Add analysis objects and initialize
RH.addAnalysis(Rivet::HZ95108());
RH.addAnalysis(ANALYSIS_HZ96215);
RH.addAnalysis("HZ98051");
RH.init();

// Loop: generate and analyze events
for ( int i = 0; i < 1000; ++i ) {
  RH.analyze(MyGenerator::generateEvent());
}

// Finish, clean up, write out histograms, ...
RH.finalize();
```

RivetHandler

```
// Create a RivetHandler given an analysis factory
RivetHandler RH(*MyHistoPackage::createFactory());

// Add analysis objects and initialize
RH.addAnalysis(Rivet::HZ95108());
RH.addAnalysis(ANALYSIS_HZ96215);
RH.addAnalysis("HZ98051");
RH.init();

// Loop: generate and analyze events
for ( int i = 0; i < 1000; ++i ) {
  RH.analyze(MyGenerator::generateEvent());
}

// Finish, clean up, write out histograms, ...
RH.finalize();
```

RivetHandler

```
// Create a RivetHandler given an analysis factory
RivetHandler RH(*MyHistoPackage::createFactory());

// Add analysis objects and initialize
RH.addAnalysis(Rivet::HZ95108());
RH.addAnalysis(ANALYSIS_HZ96215);
RH.addAnalysis("HZ98051");
RH.init();

// Loop: generate and analyze events
for ( int i = 0; i < 1000; ++i ) {
  RH.analyze(MyGenerator::generateEvent());
}

// Finish, clean up, write out histograms, ...
RH.finalize();
```

RivetHandler

```
// Create a RivetHandler given an analysis factory
RivetHandler RH(*MyHistoPackage::createFactory());

// Add analysis objects and initialize
RH.addAnalysis(Rivet::HZ95108());
RH.addAnalysis(ANALYSIS_HZ96215);
RH.addAnalysis("HZ98051");
RH.init();

// Loop: generate and analyze events
for ( int i = 0; i < 1000; ++i ) {
  RH.analyze(MyGenerator::generateEvent());
}

// Finish, clean up, write out histograms, ...
RH.finalize();
```

The `Analysis` base class

All analysis classes inherits from `Rivet::Analysis` and must override three virtual functions

- `void init();`
  For booking histograms (via the `RivetHandler`)
- `void analyze(const Rivet::Event & event);`
  For applying projections and filling histograms
- `void finalize();`
  For scaling histograms etc.
  (the `RivetHandler` will write them out)

The Analysis base class

All analysis classes inherits from `Rivet::Analysis` and must override three virtual functions

- `void init();`
  For booking histograms (via the `RivetHandler`)
- `void analyze(const Rivet::Event & event);`
  For applying projections and filling histograms
- `void finalize();`
  For scaling histograms etc.
  (the `RivetHandler` will write them out)

The Analysis base class

All analysis classes inherits from `Rivet::Analysis` and must override three virtual functions

- `void init();`
  For booking histograms (via the `RivetHandler`)
- `void analyze(const Rivet::Event & event);`
  For applying projections and filling histograms
- `void finalize();`
  For scaling histograms etc.
  (the `RivetHandler` will write them out)

Projections

The analysis class will access the desired information from the `Rivet::Event` object through `Projection` objects.

A `Projection` may project out anything from an event, eg.

- A number (such as Q^2)
- A set of final-state `Rivet::Particles`
- Jets
- ...

A projection may use other projection objects internally.

```
void HZ95108::analyze(const Event & event) {
  const DISKinematics & dk = event.applyProjection(diskin);
  const FinalStateHCM & fs = event.applyProjection(fsproj);

  histoQ2->fill(dk.Q2()/(GeV*GeV), event.weight());

  // ...

}
```

The `FinalStateHCM` projects out the final state particles in a DIS `Event` boosted to the hadronic center-of-mass frame.

Note that an `Analysis` class does not modify the `Rivet::Event` or the underlying `HepMC::GenEvent`, it will only add `Projections`.

When an `Event` is asked to apply a projection it first checks if an identical projection has been made, in which case the previous `Projection` is returned. Otherwise the projection is asked to project itself before it is returned.

Note that an `Analysis` cannot be sure that the same projection is returned as the one given as argument to `Event::applyProjection`.

Note that an Analysis class does not modify the Rivet::Event or the underlying HepMC::GenEvent, it will only add Projections.

When an Event is asked to apply a projection it first checks if an identical projection has been made, in which case the previous Projection is returned. Otherwise the projection is asked to project itself before it is returned.

Note that an Analysis cannot be sure that the same projection is returned as the one given as argument to Event::applyProjection.

Note that an Analysis class does not modify the Rivet::Event or the underlying HepMC::GenEvent, it will only add Projections.

When an Event is asked to apply a projection it first checks if an identical projection has been made, in which case the previous Projection is returned. Otherwise the projection is asked to project itself before it is returned.

Note that an Analysis cannot be sure that the same projection is returned as the one given as argument to Event::applyProjection.

The `Projection` class

A `Projection` class must override two virtual function in the base class

- `void project(const Event & e);`
  To do the actual projection.
- `int compare(const Projection & p) const;`
  To allow the `Rivet::Event` class to determine if this object is equivalent to another object of the same class. Also provides an ordering so that the applied objects can be stored in a `std::set<Projection*>` in the `Rivet::Event`.

The Projection class

A Projection class must override two virtual function in the base class

- void project(const Event & e);
  To do the actual projection.
- int compare(const Projection & p) const;
  To allow the Rivet::Event class to determine if this object is equivalent to another object of the same class. Also provides an ordering so that the applied objects can be stored in a std::set<Projection*> in the Rivet::Event.

Current Status

- A beta version will be ready before the MCnet Monte Carlo school in Durham in April.
- The general structure with projections is working.
- The interface to AIDA-compliant histograms is working.
- The interface to KTJET is working.

- Event shape classes are being added
- Analysis classes for LEP, HERA and Tevatron measurements are being added.
- A simple string-based facility to communicate cuts from an analysis to RivetGun will be replaced with something neater.
- Proper documentation is being written

The RIVET/RIVETGUN team

Andy Buckley, Jon Butterworth, Andrew Ilott, Leif Lönnblad, James Monk, Lars Sonnenschein, David Voong, Ben Waugh, . . .

http://projects.hepforge.org/rivet